Game

Jian-Jia es un joven al que le gustan los juegos. Cuando se le hace una pregunta, él prefiere ponerse a jugar en vez de responder directamente. Jian-Jia se encontró con su amiga Mei-Yu y se puso a hablar de la red de vuelos en Taiwan. Hay n ciudades en Taiwan (numeradas 0, ..., $n-1$), algunas de las cuales están conectadas por vuelos. Cada vuelo conecta dos ciudades y puede ser tomado en ambos sentidos.

Mei-Yu le pregunta a Jian-Jia si es posible ir entre cualquier par de ciudades por avión (directamente o indirectamente). Jian-Jia no quiere revelar la respuesta, en vez de esto sugirió un juego. Mei-Yu puede hacerle preguntas de la forma: '¿Están conectadas las ciudades x e y *directamente* con un vuelo?' y Jian-Jia contestará tal pregunta de manera inmediata. Mei-Yu preguntará por cada par de ciudades (exactamente una vez por cada par) dando $r = n(n-1)/2$ preguntas en total. Mei-Yu gana el juego, si después de i preguntas, para algún $i < r$, ella puede inferir si la red está conectada, es decir, si es posible viajar entre cualquier par de ciudades por vuelos, ya sea directamente o indirectamente. En caso contrario, es decir, si Mei-Yu necesita hacer todas las r preguntas, entonces Jin-Jia gana el juego.

Con la finalidad de hacer que el juego sea más divertido para Jian-Jia, los amigos acuerdan que Jian-Jia puede olvidarse de la red real de vuelos e inventar la red conforme el juego progrese, escogiendo sus respuesta basandose en las preguntas previas de Mei-Yu. Tu tarea es ayudar a Jian-Jia a ganar el juego, decidiendo cómo debería responder las preguntas.

Ejemplos

Explicaremos las reglas del juego con tres ejemplos. Cada ejemplo tiene $n = 4$ ciudades y, por lo tanto, $r = 6$ rondas de preguntas y respuestas.

En el primer ejemplo (la siguiente tabla), Jian-Jia *pierde* porque después de la ronda 4, Mei-Yu sabe con certeza que uno puede viajar entre cualquier par de ciudades por vuelos, sin importar como Jian-Jia reponda las preguntas 5 o 6.

ronda	pregunta	respuesta
1	0, 1	si
2	3, 0	si
3	1, 2	no
4	0, 2	si
-----	--------	------
5	3, 1	no
6	2, 3	no

En el siguiente ejemplo, Mei-Yu puede probar que después de la ronda 3, sin importar como responda

Jian-Jia las preguntas 4, 5 o 6, *no se puede viajar* entre las ciudades 0 y 1, por lo tanto Jian-Jia pierde nuevamente.

ronda	pregunta	respuesta
1	0, 3	no
2	2, 0	no
3	0, 1	no
-----	--------	------
4	1, 2	si
5	1, 3	si
6	2, 3	si

En el ejemplo final, Mei-Yu *no puede determinar con certeza* si es posible viajar entre dos ciudades cualesquiera hasta que estén respondidas todas las seis preguntas, por lo que Jian-Jia *gana* el juego. Específicamente, como Jian-Jia respondió *si* a la última pregunta (en la siguiente tabla), entonces es posible viajar entre cualquier par de ciudades. Sin embargo, si Jian-Jia hubiera respondido *no* en lugar de *si* a la última pregunta entonces sería imposible.

ronda	pregunta	respuesta
1	0, 3	no
2	1, 0	si
3	0, 2	no
4	3, 1	si
5	1, 2	no
6	2, 3	si

Tarea

Por favor, escribe un programa que ayude a Jian-Jia a ganar el juego. Note que ni Mei-Yu ni Jian-Jia conocen la estrategia del otro. Mei-Yu puede preguntar por pares de ciudades en cualquier orden y Jian-Jia debe reponder inmediatamente sin saber las preguntas siguientes. Debes implementar las siguientes dos funciones.

- `initialize(n)` -- Llamaremos primero a la función `initialize`. El paramétro n es el número de ciudades.
- `hasEdge(u, v)` -- Luego llamaremos $r = n(n-1)/2$ veces a la función `hasEdge`. Esas llamadas representan las preguntas de Mei-Yu, en el orden en que ella las hace. Debes responder si hay un vuelo directo entra las ciudades u y v. Específicamente, el valor de retorno debe ser 1 si hay un vuelo directo, y 0 en otro caso.

Subtareas

Cada subtarea consiste de varios juegos. Únicamente obtendrás puntos por una subtarea si tu programa hace que Jian-Jia gane todos los juegos de esa subtarea.

subtarea	puntos	n
1	15	$n = 4$
2	27	$4 \leq n \leq 80$
3	58	$4 \leq n \leq 1500$

Detalles de la implementación

Debes enviar exactamente un archivo, llamado `game.c`, `game.cpp` o `game.pas`. Este archivo implementa los subprogramas descritos anteriormente usando las siguientes firmas.

Programas C/C++

```
void initialize(int n);
int hasEdge(int u, int v);
```

Programas Pascal

```
procedure initialize(n: longint);
function hasEdge(u, v: longint): longint;
```

Sample grader

El *grader* de ejemplo lee el input en el siguiente formato:

- línea 1: `n`
- las siguientes r líneas: cada línea contiene dos enteros `u` y `v` que describen una pregunta respecto a las ciudades u y v.